REPAST MODEL TESTING GUIDE

JONATHAN OZIK, NICK COLLIER - REPAST DEVELOPMENT TEAM

0. Before we Get Started

Before we can do anything with Repast Simphony, we need to make sure that we have a proper installation of Repast Simphony 2.1. Instructions on downloading and installing Repast Simphony on various platforms can be found on the Repast website.

1. Getting Started with Repast Simphony Model Testing

This guide will walk you through a few model testing use cases using Repast Simphony. To learn more about model testing, including the benefits of test driven development (TDD) when developing agent models, see Collier and Ozik (2013)[1]. For more information on the JUnit testing framework that we will be using, see http://junit.org.

To add tests into an existing Repast Simphony project, we recommend the following setup steps:

(1) Add a `test` source folder to the project. This can be accomplished in a number of ways. One way is to right click on the project and navigate to *New* → *Source Folder* (Fig. 1). Then fill in `test` in the *Folder name* text field and click on *Finish* (Fig. 2).

(2) Modify the output folder for the `test` source folder. Right click on the project and navigate to *Properties*. Then choose the *Java Build Path* entry in the left bar and select the *Source* tab. Select the check box that reads *Allow output folders for source folders* and expand the `test` entry (Fig. 3). Select the *Output folder* entry and click on the *Edit* button (Fig. 4). Then select the *Specific output folder* option and input `testbin`, or something else different from `bin` (Fig. 5). Click on *Okay* and then *Okay* again.

(3) Add a JUnit Test Case by right clicking on the project in the Package Explorer view, choosing *New* → *Other...* (Fig. 6). Then navigate to *Java* → *JUnit* and choose *Junit Test Case* (Fig. 7). In the *New JUnit Test Case* wizard choose the *New JUnit 4 test* option, specify the correct source folder (`test`), name your test case, include all the method stubs, and click on *Finish* (Fig. 8). If JUnit was not previously on your project's build path, you will see a dialog asking if you'd like to add it (Fig. 9). Click *OK* to add the JUnit 4 library to the project's build path.

Date: June 21, 2013.

[1]Collier, N, and J Ozik. Test-Driven Agent-Based Simulation Development. To appear in WSC 2013 Proceedings. Washington, D.C., 2013.

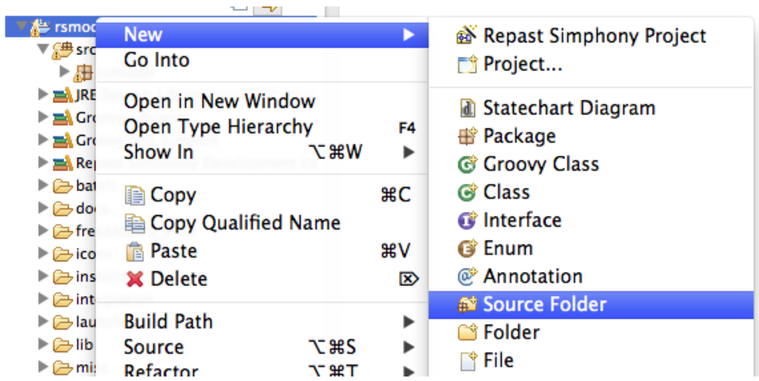
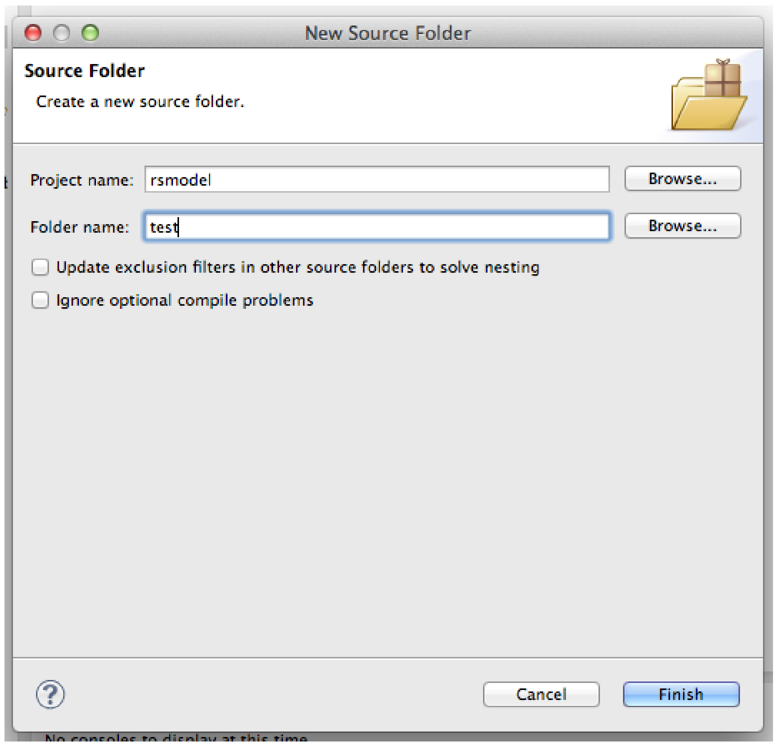


FIGURE 1. Selecting your project, right-clicking and choosing *New* → *Source Folder*.


New Source Folder

Source Folder

Create a new source folder.

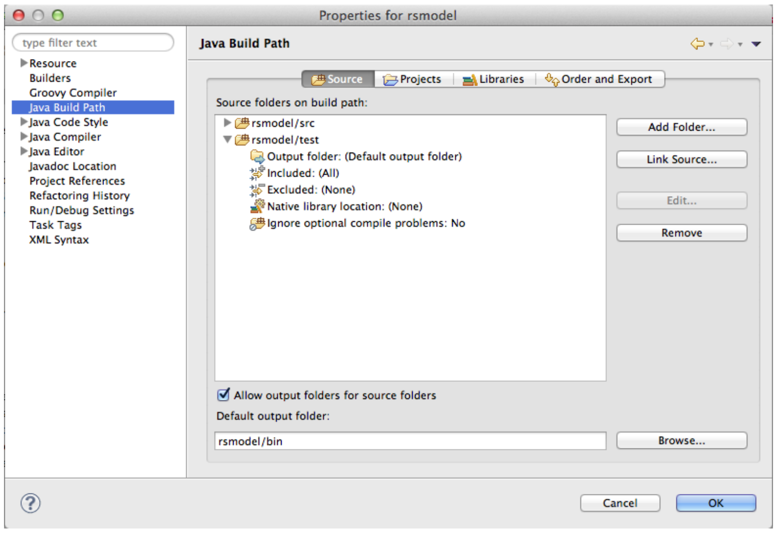
Project name: rsmodel

Folder name: test

☐ Update exclusion filters in other source folders to solve nesting

☐ Ignore optional compile problems

FIGURE 2. New source folder wizard.



Source folders on build path:

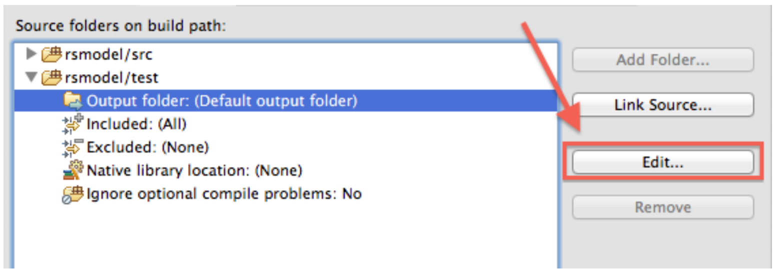
- rsmodel/src
- rsmodel/test
  - Output folder: (Default output folder)
  - Included: (All)
  - Excluded: (None)
  - Native library location: (None)
  - Ignore optional compile problems: No

☑ Allow output folders for source folders

Default output folder:

rsmodel/bin

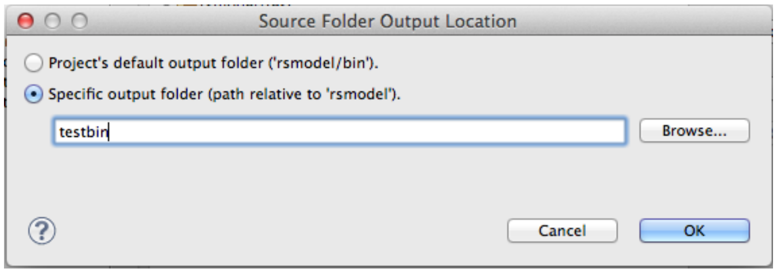
FIGURE 3. *Java Build Path* → *Source* tab in a project's properties.



Source folders on build path:

- rsmodel/src
- rsmodel/test
  - Output folder: (Default output folder)
  - Included: (All)
  - Excluded: (None)
  - Native library location: (None)
  - Ignore optional compile problems: No

FIGURE 4. Edit the output folder of the `test` source folder.


Source Folder Output Location

Project's default output folder ('rsmodel/bin').
[✓ Specific output folder (path relative to 'rsmodel').]

testbin

FIGURE 5. Choosing the source folder output location.

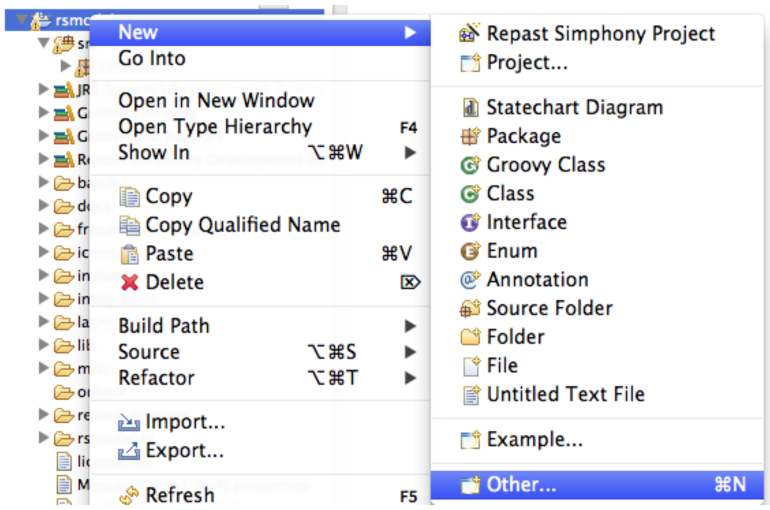


FIGURE 6. Selecting your project, right-clicking and choosing New → Other... .

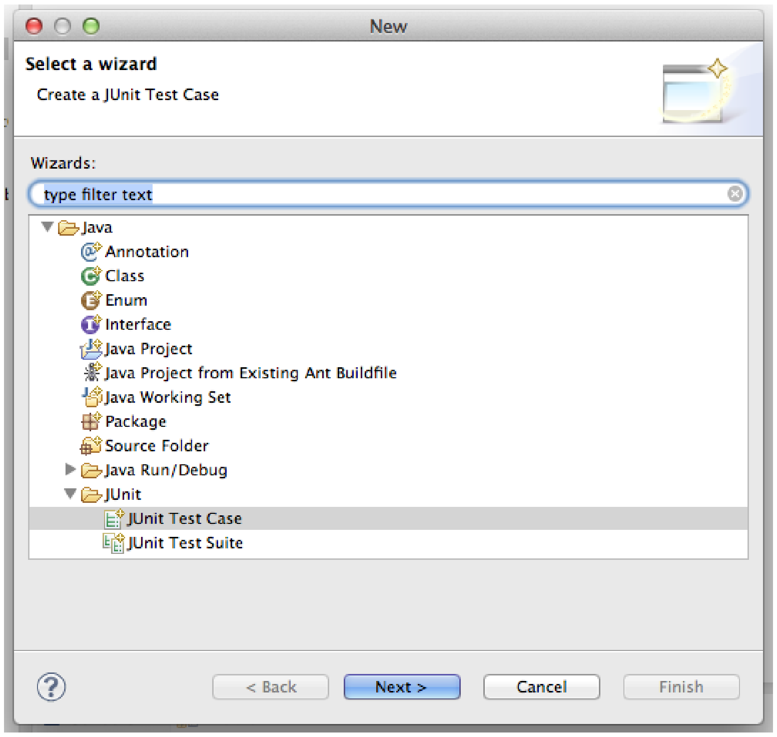
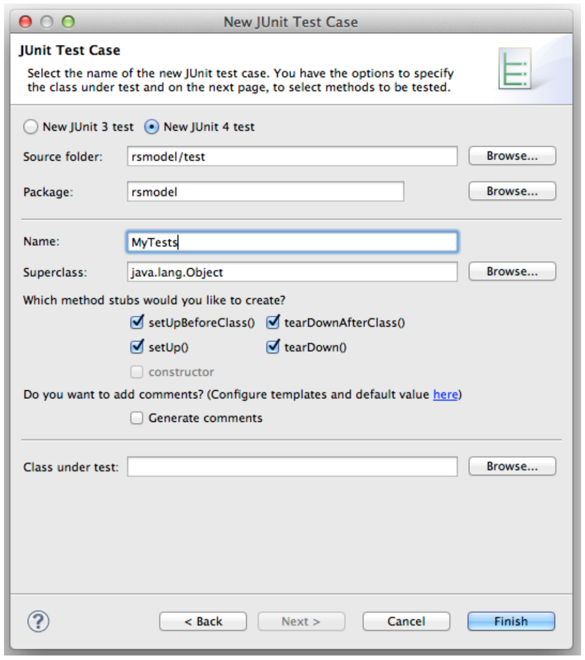


FIGURE 7. The JUnit Test Case option within Java → JUnit.


New JUnit Test Case

JUnit Test Case

Select the name of the new JUnit test case. You have the options to specify the class under test and on the next page, to select methods to be tested.

New JUnit 3 test [✓ New JUnit 4 test]

Source folder: rsmodel/test

Package: rsmodel

Name: MyTests

Superclass: java.lang.Object

Which method stubs would you like to create?

✓ setUpBeforeClass() ✓ tearDownAfterClass()

✓ setUp() ✓ tearDown()

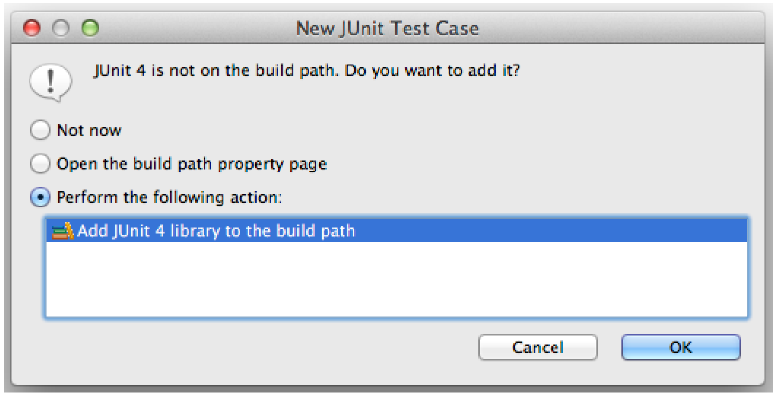
constructor

Do you want to add comments? (Configure templates and default value here)

Generate comments

Class under test:

FIGURE 8. The new *JUnit Test Case* wizard.


New JUnit Test Case

JUnit 4 is not on the build path. Do you want to add it?

Not now

Open the build path property page

[✓ Perform the following action:]

Add JUnit 4 library to the build path

FIGURE 9. Add JUnit 4 library to build path.

Once the setup is complete, there are a number of different types of model tests that can be created, based on the nature of the model behavior that is being tested. We will go over a few such cases next.

1.1. **Use Case 1: Simple Unit Testing with Repast Simphony Models.**

If the elements being tested are relatively decoupled, there is nothing special that needs to be done in terms of test case setup. In this scenario, the @BeforeClass, @AfterClass, @Before, and @After annotated methods do not need any Repast Simphony specific elements and tests can be written in the usual way JUnit tests are written.

1.2. **Use Case 2: Schedule Based Model Testing with Repast Simphony Models.**

Some model behaviors that you might want to test involve scheduled behaviors. For example, you might want to know that at a certain tick, specific model actions had occurred. This is especially relevant to any Statecharts related behaviors. In this situation, the RunEnvironment will nee to be set up with a Schedule object. Listing 1 shows an example where the setup is executed before each of the tests that will be run.

```
1  @Before
2  public void setUp() throws Exception {
3      Schedule schedule = new Schedule();
4      RunEnvironment.init(schedule, null, null, true);
5      Context context = new DefaultContext();
6      RunState.init().setMasterContext(context);
7
8      // Any additional setup
9  }
```

Listing 1. @Setup method in a schedule dependent test case.

Line 3 shows the creation of the Schedule object. It is then sent as the first parameter to the RunEnvironment static `init` method. The second and third parameters can be `null` for this case and the fourth parameter indicates whether this is a batch (i.e., headless) run, which it is so we specify `true`. Line 5 creates a new DefaultContext that is set as the master context of the `init`-ed RunState in line 6. Lines 5 and 6 are only strictly necessary for testing behaviors that rely on a master context being set, which is the case for Statecharts.

At this point, tests can be written such as in Listing 2:

```java
@Test
public void testColonizedToInfected() {
   ISchedule schedule = RunEnvironment.getInstance().
         getCurrentSchedule();
   Person p = new Person();
   assertEquals(UNINFECTED, p.getStatus());
   for (int i = 0; i < 5; ++i) {
      schedule.execute();
   }
   assertEquals(INFECTED, p.getStatus());
}
```

Listing 2. @Setup method in a schedule dependent test case.

In this hypothetical example, a Person object is created, it is verified that the person's status is `UNINFECTED`, the schedule is advanced 5 times, at which point the person's status is checked to see that it is `INFECTED`. An important item to note here is that the scheduler in Repast Simphony doesn't just allow for discrete time steps so the fact that the schedule is executed 5 times doesn't necessarily mean that we will find ourselves at tick 5 after the `for` loop, unless there were actions scheduled only to occur on every tick.

1.3. **Use Case 3: Context Builder Based Model Testing with Repast Simphony Models.** For cases where the specific setup defined in a ContextBuilder is required,

```java
. . .
public Context context;

@Before
public void setUp() throws Exception {
   Schedule schedule = new Schedule();
   RunEnvironment.init(schedule, null, null, true);
   context = new DefaultContext();
   MyContextBuilder builder = new MyContextBuilder();
   context = builder.build(context);
   RunState.init().setMasterContext(context);

   // Any additional setup
}
. . .
```

Listing 3. @Setup method in a schedule dependent test case.

1.4. **Use Case 4: Model Testing with ReLogo Models.** For cases

```
. . .
static UserObserver observer;

@BeforeClass
public static void setUpBeforeClass() throws Exception {
   String scenarioDirString = "ModelName.rs"
   ScenarioUtils.setScenarioDir(new File(scenarioDirString))
   File paramsFile = new File(ScenarioUtils.getScenarioDir(), "parameters.xml");

   ParametersParser pp = new ParametersParser(paramsFile);
   Parameters params = pp.getParameters();
   RunEnvironment.init(new Schedule(), null, params, true);
   Context context = new DefaultContext();
   SimBuilder builder = new SimBuilder();
   context = builder.build(context);
   observer = (UserObserver)context.iterator().next();

   // Any additional before class setup
}

@Before
public void setUp() throws Exception {
   observer.clearAll();

   // Any additional setup
}
. . .
```

Listing 4. @Setup method in a schedule dependent test case.